Advanced Programming

Philip Hanna CSC2021

Games Programming

Philip Hanna CSC2022

Module Handbook

Welcome:

Welcome to CSC2021 Advanced Programming using Android and C++ and CSC2022 Games Programming using Android and C++.

This co-taught module will provide an intensive but hopefully fun means of improving your programming skills. My goal is to help you become a more professional and capable programmer who is well prepared to undertake work experience in a development studio.

This handbook contains lots of useful information on how the module is structured to achieve the aims mentioned above. You will also find useful links and contact details if you require further information or want to get in contact with me.

Philip Hanna

Contents:

Learning Objectives

What does this module aim to accomplish?

All modules have aims[1], i.e. things they hope to accomplish. In the welcome to this handbook I indicated that I hope to help you improve your programming skills and become a more accomplished programmer.

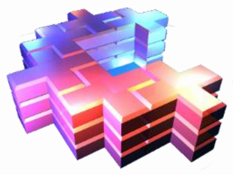
Primarily we will accomplish this by developing an Android game and delving into 2D game development within the lectures. Other than being fun, game development is one of the better ways of improving programming skills as it gives us a context within which to explore core computer science and software engineering topics, e.g. algorithmic speed and efficiency, object oriented modelling, graphical manipulation, artificial intelligence, etc.

Developing an Android game is not an easy undertaking – it requires a lot of time and effort, hence, groups of four-to-five students will team up to develop their game. Other than making the task more manageable, this will also help you prepare for the team-based environment you will enter in your placement year.

So, how does C++ factor into all of this? It's in the module title after all. We will explore C++ as a means of helping you write better code, be this in Java, C#, C++, etc.

C++ is a long established language and is widely used today to write 'high-performance' code. C++ was 'born' in 1983 and is older than most people taking this course. Because of its age, it is an 'old-school' language. This is a rather polite way of saying that it is an unfriendly and utterly unforgiving language (certainly compared to most modern programming languages).

However, it will enable us to expose how managed languages, such as Java, set out data in memory and execute code. In doing this, we will be able to avoid a wide range of common situations where a Java programmer might inadvertently write some code that imposes a massive performance hit 'behind the scenes'. If this is not enough for you, then you'll also be able to impress employers in interviews with talk of in-lining, dynamic memory (de-) allocation, pointer arithmetic, etc.



The rest of this section is based around the exploration of the above aims. In particular, we will explore: how we can go about improving programming skills; and, how the improvement will be assessed and mapped onto a mark for the module.

[1] 'Aims' or should that be 'Learning Outcomes'? Does it really matter to you? If it does, don't worry we'll get onto what I hope you'll be able to demonstrate following the course in the next section.

Learning on this module

In this section I've tried to outline my thinking behind how I've structured the module to help you improve your programming skills. In this sense, this section also contains my honest advice of how to get most out of the project and maximise your overall mark.

The 'philosophy' behind my approach can be summed up rather nicely in a short exercise that Professor Phil Race (a well-known educationalist) carried out during a workshop at QUB. He asked all the participants to write down one activity or thing they learnt to do – it could be anything from juggling, to learning a foreign language, to playing a musical instrument, to driving an off-road vehicle, etc. He then asked us to write the reasons why we wanted to learn the activity or thing. Finally, he asked us to write down how we went about learning the activity or thing.

As might be expected a wide range of different types of activity were reported by the participants. However, when we got around to reading out the reasons why we learnt the activity there were really only two different reasons: it was either useful or fun. The manner of how we learnt the activity was even more stark: we did the activity and got better through practice. In other words, people want to learn things because they are either useful or fun, and people learn things by doing them and getting better through practice.

Ok, this is hardly surprising[2]; however, it does provide me with a challenge as the aim of this course is to improve your programming skills. I've got to provide you with something you want to do and something you feel is useful, whilst at the same time structuring the course in a manner that facilitates the practice of programming.

How do I intend to do this? Well, mostly by giving you the freedom to do something that you want to do!

Now admittedly, as the project is about creating a game, as opposed to say axiomatic semantics, I'm starting from somewhat of a privileged position. My particular approach is to let you decide on the game that you wish to develop. My only requirement is that it should be something that you find interesting. It might be an adaptation of a classic video game, a variant of a currently popular game, or an entirely novel idea. You can aim to write something that will make you millions, or reach out to a demographic not normally interested in current genres of game. Whatever floats your boat. As an aid towards selecting a project, I hope to spend the first couple of weeks giving you lots of ideas and offering some advice in terms of what can be done within the available time.

[2] Do you believe our preferred adult forms of learning are somewhat similar to that of children (and indeed of other primates)? As individuals who have got through a number of years of education, how would you assess primary, secondary and tertiary level education in terms of the extent to which it engages and excites individuals (i.e. is it perceived as useful and fun) and provides opportunity for skills to be put into practice?

Apart from selecting the game topic, you will also be able to decide how adventurous you wish to be in terms of your programming. It you feel like pushing the boat out, then you can opt for a more complex game, or alternatively, you may decide to focus on something more straightforward. As with the game topic, I will offer lots of advice in terms of clearly defining the mapping from game complexity onto the final awarded mark.

You will also be given an input into how marks will be distributed for your project – i.e. if you wish to develop a puzzle game that places more focus on artificial intelligence, then you can increase the number of marks in this category. Finally, you can also decide who you wish to work with when tackling the project. More information on the project is provided later.

Our learning contract

I have taken the liberty of setting out a 'learning agreement' which defines the things you can expect of me and also the things expected of you. 'Expected', in this context, entails that as long as we both hold to the items within the agreement then there is an excellent chance that you will get a good mark in the project.

Learning Agreement

Things which you can expect of me:

In general:

- I'm here to help – and will very happily provide assistance and direction if requested (obviously within reason!).
- I will be supportive and non-judgemental – everyone learns at a difference pace and making mistakes is very often the best way to learn.

Within lectures:

- I won't spend lots of time regurgitating material covered elsewhere within the notes.
- I'm more than happy to answer questions within lectures; if something is unclear please say.

For assessment:

- I will be fair, honest and consistent when assessing projects. If the stated reason for an awarded mark is unclear, please do ask for clarification.

Things which I will expect of you:

- You will attend lectures and explore the directed reading (again, within reason – there will be weeks where you have a dental appointment, or a number of due assignments, etc.).
- You will work on your project throughout the duration of the module, and you won't leave it until the last couple of weeks. Students who have failed to work consistently, in this module and other project based modules, end up pulling a number of hellish all-nighters at a period where most other modules also have their final assignment hand-in date, i.e. everything suffers. This is most certainly not in your interest.

Module Structure and Timeline

How is this module structured?

The module is 100% project based and how you do within the project will determine your mark for the module. As an aside, I'm not a fan of final written examinations so I don't use them on any of my modules (hurrah!); however, it does mean that the course work you submit as part of the module will determine your entire module mark. As this module is worth 1.5 module credits this means the project is extremely important (you cannot enter into third year until this module has been passed).

Unsurprisingly, lecture content within the module is firmly based around the development of your project. A complete set of lecture notes (with audio narration) will be made available on Queen's Online and YouTube.

Lectures will not be entirely 'traditional' – whatever that means. In particular, I won't regurgitate the lectures notes within the lectures; using a lecture to provide a set of notes is a rather poor use of an hour (besides, you already have them on QoL!). Instead, lectures will hopefully provide information and advice that will help you with your project. To this end, lectures will extensively link to other resources, many online, that you can explore as and when needed.

Later in the module you will also be able to determine what is explored within lectures. In particular, I will ask you to identify topics you would like covered in the next week. This means I can explore in the lectures the collective areas that are of most relevance and importance to the projects.

The next page provides a breakdown of the topics covered in the module.

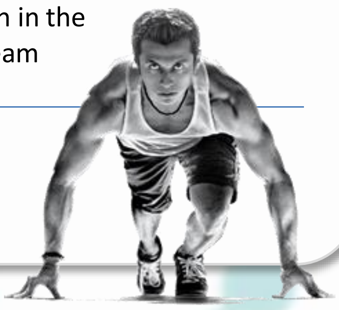
Weeks 1-3: Getting setup and ready to go

The first few weeks are all about setting out the aims of the module and getting you ready to undertake the project (teams setup, how to work effectively in a team and plan your time, basics of Android programming covered).

Week 1 (Semester 1)	**Introduction**	
Lecture 1: Module introduction	**Lecture 2:** Project overview	**Lecture 3:** Getting setup
Overview of the learning aims and how you can best succeed on this module.	Overview of the project and associated activities you will undertake in the module. Details of how your efforts and accomplishments will translate into a mark for the module.	Details of how you can setup an appropriate development environment for the module, including installing Eclipse, Android SDK and SVN. Details of tablet and emulator setup/usage on this module.
Project:	**Getting setup**	
Get your development environment setup.		

Week 2 (Semester 1)	**Team formation**	
Lecture 1: Project management	**Lecture 2:** Agile and Lean software development	**Lecture 3:** Android programming 101
Guest lecture from Citi on how best to manage a project.	Exploration of key agile and lean software development principles and how to make use of them in your project.	Key introductory principles behind Android development and creating and deploying your first Android app.
Project:	Getting started	
Establish your team and explore the type of project you would like to develop. Creating your first Android app.		

Week 3 (Semester 1)	**Working as a team**	
Lecture 1: Communication and Teamwork	**Lecture 2**: Team Working / Architecture Design	**Lecture 3**: SVN / Android Programming 101+
Guest lecture from Citi on how best to work effectively as a team and communicate well.	Team working exercise. Exploration of the software architectures that are used in games development and how you can plan your game.	Using SVN to enable your team to collaboratively develop software. Developing an app suitable for holding a game.
Project:	Sprint 1 [Getting up to speed with Android]: Planning	
Adding detail to your game design and planning what you want to accomplish in the first sprint. Creating a foundational app for your game. Submission of your team and project goals document.		



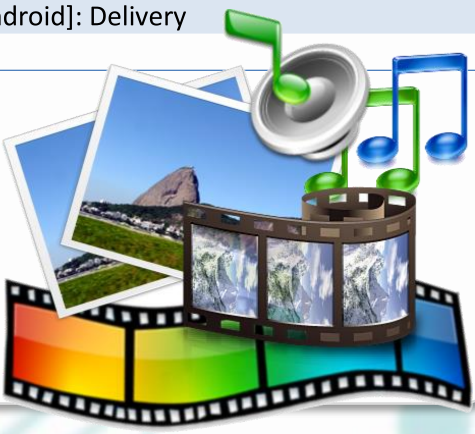
Weeks 4-6: Your first sprint

You will be using an agile software development approach to build your project, with development split up into a number of short planning and implementation phases, called sprints. Your first sprint will get you to implement a number of core game programming components (input, graphics and sound) covered in the lectures.

Week 4 (Semester 1)	**Key game programming concepts 1**	
Lecture 1: The game loop	**Lecture 2**: Role of graphics in games	**Lecture 3**: Alpha blending and text/bitmap display
Exploration of the update/render loop within games and integrating this within the app lifecycle.	The role of graphics within (2D) games. Loading and displaying bitmaps (including loading assets in general).	Blending images together to achieve different visual effects. Handling text within games.
Project:	Sprint 1 [Getting up to speed with Android]: Delivery	
Commencement of sprint 1 team objectives.		

Week 5 (Semester 1)	**Key game programming concepts 2**	
Lecture 1: Getting input into games	**Lecture 2**: Graphical transforms	**Lecture 3**: Sound and music within games
How to get and use different types of input within your game (touch events, key events, accelerometer, etc.).	Introduction to different types of affine graphical transform (translation, scale, rotation, shear) and their potential use within your game.	How to load and use sound effects and stream music within your game.
Project:	Sprint 1 [Getting up to speed with Android]: Delivery	
Progression of sprint 1 team objectives.		

Week 6 (Semester 1)	**Key game programming concepts 3**	
Lecture 1: Viewports	**Lecture 2**: Animation	**Lecture 3**: Image ribbons and tiles
Handling different screen sizes and displaying windows into a virtual game world.	Displaying a series of successive images to animate a game object.	Combining seamless images within a ribbon or tile structure to bring about scrolling backgrounds.
Project:	Sprint 1 [Getting up to speed with Android]: Delivery	
Completion of sprint 1 team objectives.		



Weeks 7-9: Building your foundation

Things start to come together in the second sprint. The lectures in week 7 will look at how the different items of functionality can be combined together into a coherent architecture. Your goal within the second sprint will be to combine the various code fragments, developed in the first sprint, into your own architecture (an architecture that will be used to build the rest of your game in subsequent sprints).

Whilst week 8 is free from lectures to enable you to work on building the foundation of your game, week 9 lectures resume the coverage of new content (looking at AI and how you can bring your game to 'life').

Week 7 (Semester 1)	**Game architecture**	
Lecture 1: Game screen management	**Lecture 2**: Creating a coherent architecture	**Lecture 3**: Practical advice
Designing and developing your game around a number of separate game screens.	Linking together assets, input, audio, graphics and game screens within a coherent game architecture.	Practical advice on object design, update and rendering.
Project:	Sprint 1 review. Sprint 2 [Developing your game architecture]: Planning	
Reflecting as a team on how sprint one went. Planning what you want to accomplish in the second sprint.		

Week 8 (Semester 1)	**Dev time!**
No lectures this week. It's time to pull together everything learnt in the last seven weeks.	
Project:	Sprint 2 [Developing your game architecture]: Delivery
Commencement of sprint 2 team objectives.	

Week 9 (Semester 1)	**Artificial Intelligence 1**	
Lecture 1: Overview of AI	**Lecture 2**: Steering behaviours 1	**Lecture 3**: Steering behaviours 2
Exploration of AI within games and outline of a typical AI engine.	Primitive forms of AI steering behaviour (seek, flee, align, arrive, etc.).	More sophisticated forms of steering behaviour (wonder, interpose, intercept, separate, etc.).
Project:	Sprint 2 [Developing your game architecture]: Delivery	
Completion of sprint 2 team objectives.		

Weeks 10-12: The heart of your game

The final sprint leading up to the end of the first semester will provide you with an opportunity to bring to life the core elements of your game (albeit, in a likely unrefined manner). By the end of the first semester you will have a basic, but playable, version of your game created. The lectures in the final weeks (from week 9) will help support this by exploring a range of algorithms and approaches that can be used to enable game objects to move about and interact with one another.

Week 10 (Semester 1)	**Team review**	
Lecture 1: Conflict Resolution	**Lecture 2**: Review of progress and team working	**Lecture 3**: Collision detection and avoidance
Guest lecture from Citi on how best to handle conflict within a team and work well together.	Reviewing team progress to date and mapping what has been developed onto the marking scheme.	Exploration of collision detection between 2D shape primitives and how collisions can be handed or avoided by game objects.
Project:	Sprint 2 review. Sprint 3 [The heart of your game]: Planning.	
Reflecting as a team on how sprint two went. Planning what you want to accomplish in the third sprint. Submission of project progress report.		

Week 11 (Semester 1)	**Artificial Intelligence 2**	
Lecture 1: Game agents and decision making	**Lecture 2**: Path finding and following / Jumping	**Lecture 3**: Aiming
Notion of an autonomous agent within a game: how it can ‘sense’ its world and make decisions.	Details of how AI agents can build and then following a path to a target destination. Consideration of what needs to be put in place if agents can safely jump.	Coverage of how agents can launch projectiles towards some target position or work out if they are likely to get hit.
Project:	Sprint 3 [The heart of your game]: Delivery.	
Commencement of sprint 3 team objectives.		

Week 12 (Semester 1)	**Semester round up.**	
Lecture 1: Particle systems / Strategic thinking	**Lecture 2**: Semester review / Prize ceremony	**No lecture**.
Overview of particle systems in games. Strategic thinking in agents and board game AI.	Review of progress to date (and award of prizes). Exploration of potential activities for the semester break.	
Project:		
Completion of sprint 3 team objectives.		

Weeks 1-3 (Semester 2): Adding ‘wow’ to your game

The start of the second semester represents an important evolution in your project. The core concepts and approaches needed to realise your game have already been covered in the first semester and you will have developed a basic version of your game. In the six weeks that remain you will be adding the ‘wow’ factor to your game, adding more complex features and refining your existing functionality.

Lectures in the second semester will explore C++ and UX design – important areas that can be used to help improve the performance and experience of your game.

Week 1 (Semester 2)	**Introduction to C++**	
Lecture 1: Semester planning / Intro to C++	**Lecture 2**: C++ Compilation process	**Lecture 3**: C++ Data types
Overview of development activities and coding challenges within this semester. Overview of C++.	How code gets to be compiled in programming languages. Differences between C++ and Java.	Exploration of different primitive data types in C++
Project:	Sprint 3 review. Sprint 4 [Adding some meat]: Planning.	
Reflecting as a team on how sprint three went. Planning what you want to accomplish in the fourth sprint. Submission of project progress report.		

Week 2 (Semester 2)	**C++ and UX Design 1**	
Lecture 1: C++ Memory and pointers	**Lecture 2**: UX Design - Overview and design principles	**Lecture 3**: Project Support
Details of how primitive data is stored in memory and how pointers provide a powerful means of access.	Introduction to user experience design and why this matters to you.	You will get to decide what topics/areas should be covered in this lecture to help you with your project.
Project:	Sprint 4 [Adding some meat]: Delivery.	
Commencement of sprint 4 team objectives.		

Week 3 (Semester 2)	**C++ and UX Design 2**	
Lecture 1: C++ Dynamic memory allocation / C++ Classes 1	**Lecture 2**: UX Design - How we read and what we see	**Lecture 3:** Project Support
How memory can be dynamically allocated in C++. Introduction to C++ classes and how they differ (slightly but importantly) from Java.	Exploring how we scan and read information and what we tend to focus on (also what we tend not to see).	You will get to decide what topics/areas should be covered in this lecture to help you with your project.
Project:	Sprint 4 [Adding some meat]: Delivery.	
Completion of sprint 4 team objectives.		

Weeks 4-7 (Semester 2): Finishing your game

In the final sprint you will add whatever finishing touches are needed to your game and submit at the start of Week 7. The lectures will conclude with contact time thereafter used to directly support your project.

Week 4 (Semester 2)	**C++ and UX Design 3**	
Lecture 1: C++ Classes 2	**Lecture 2**: UX Design - How we think	**Lecture 3**:Project Support
More details on classes in C++ (copy, assignment and move constructors, rule of 3, this keyword and casting).	Consideration of how we tend to think and exploration of how UX design principles might apply to your game.	You will get to decide what topics/areas should be covered in this lecture to help you with your project.
Project:	Sprint 4 review. Sprint 5 [Applying the polish]: Planning.	
Reflecting as a team on how sprint four went. Planning what you want to accomplish in the fifth sprint. Submission of project progress report.		

Week 5 (Semester 2)	**Final slice of C++**	
Lecture 1: C++ Optimisation techniques 1	**Lecture 2**: C++ Optimisation techniques 2	**Lecture 3**: Round-up and Assessment Details
Advice on how code in C++ and other languages (including Java) can be optimised.	More advice on how code in C++ and other languages can be optimised.	The formal lectures come to an end with precise details given on the assessment schedule in Week 7.
Project:	Sprint 5 [Applying the polish]: Delivery.	
Commencement of sprint 5 team objectives.		

Week 6 (Semester 2)	**The final push**	
Lecture 1: Project Support	**Lecture 2**: Project Support	**Lecture 3**: Project support
Lectures in the final week will be based around supporting the projects		
Project:	Sprint 5 [Applying the polish]: Delivery.	
Completion of sprint 5 team objectives.		

Week 7 (Semester 2)	**Assessment**
No lectures this week. Your project will be submitted and assessed.	
Project:	Submission and Assessment.

Project Overview

The project – a game you really want to develop

The most important part of this module is the development of a game that highlights your programming skills. You will be able to select the game and will also have control in terms of how marks are allocated within the assessment. I will provide advice on what you might like to tackle and the boundaries you will wish to impose on the development. The module contains a number of hand-in points that provide you with an opportunity to report progress and get feedback. Your project will be formally assessed at the end of the module (this can involve a short viva).

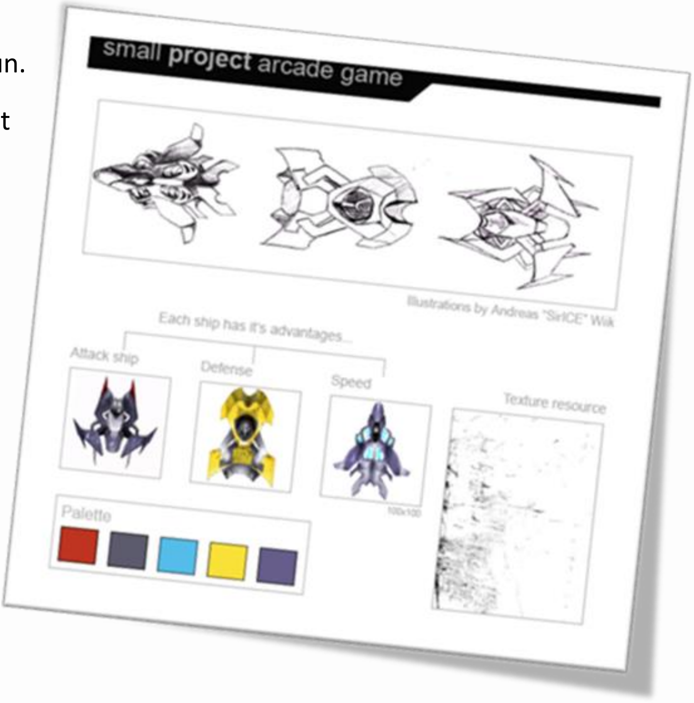
Selecting a project:

Lectures and directed reading in the first two weeks of the course will explore different types and genre of game. Hopefully this coverage will provide you with some ideas and enable you to select a project that you find really interesting. It might be an adaptation of a classic video game, a variant on a popular current game, a computer version of a traditional board game, or an entirely novel idea.

Obviously there are constraints in terms of what can be done: you won't have two years and a budget of some millions to form a large team of programmers, animators, artists, etc. to realise your vision of a game. You will have a period of eighteen weeks, a budget that might stretch to include some late night caffeine, and a team of, well, yourself and some of your fellow students to complete the game. As an aside, this is not that dissimilar to the early days of 8-bit game development or many of the current start-up mobile game developers.

My advice for selecting a project is as follows:

- Firstly, and most importantly, plan to develop something that you think is fun.
- Base your project on a simple idea that you feel can be easily implemented within the available time.
- Plan to develop something along the lines of a short demo, as opposed to a fully featured game. Don't develop something that will take ten hours of play to see all of the features, instead pick something that can be experienced within ten minute of play.


small project arcade game
Illustrations by Andreas "SirICE" Wik
Each ship has it's advantages...
Attack ship
Defense
Speed
Texture resource
100x100
Palette

- Include contingency planning within your project. Decide what will be the core features of your game and plan to implement these first. Any other additional bells and whistles can be added later. Not including a bell or whistle is not an issue, failing to include a key game feature is!
- Carefully read the assessment criteria for the project as this will direct your thinking towards a project that hits upon the areas for which marks are awarded (although you will have some say in terms of how the marks will be distributed).

At the end of the third week you will submit a short outline of your planned game and its features (more on this shortly). Each submission will be reviewed with the aim of giving you some feedback and advice in terms of the challenge and scope of your project.

Working as a team:

You will work as part of a small team on the game project (working in a team helps makes the project more manageable and also improves learning). Teams will normally be limited to a maximum of five members, with three as the minimum size. I will be happy to act as a brokering service in terms of forming groups if needed.

Working as a team does require effort in terms of coordinating work between team members and also integrating the various pieces of developed code. We'll explore how this can be done within a number of the lectures. At times you will need to show flexibility in how you work with others whose approach to software development differs from your own. However, a team that 'gels' offers the possibility of developing a very impressive game!

In terms of the project assessment, a single mark will be awarded to the entire project. An individual mark will then be derived for each team member based on their specific contribution to the project (please see the project assessment document for more information). The assessment has been structured to take into account the size of the team, i.e. working in a team of three will not be disadvantageous. Each team member should plan to spend about six-to-eight hours each week working on the project.

Developing your project using the Android SDK:

You will certainly be familiar with the Java programming language. However, you might not be familiar with Android programming – whilst it is based on Java, developing for Android involves getting familiar with a mobile platform and the rich SDK that goes along with this.

Now, you might want to put down any sharp objects before reading the next sentence. Whilst you will use the Android SDK to develop your project, this module will **not** contain a series of lectures or practicals on how to develop Android apps. Instead, it will be your responsibility to pick up and learn this (I'll provide links to online resources, textbooks, exercises, etc.). In this regard, the module differs from Stage 1 modules which typically have a series of guided practical exercises.

There are a couple of very good reasons why I have adopted this position.

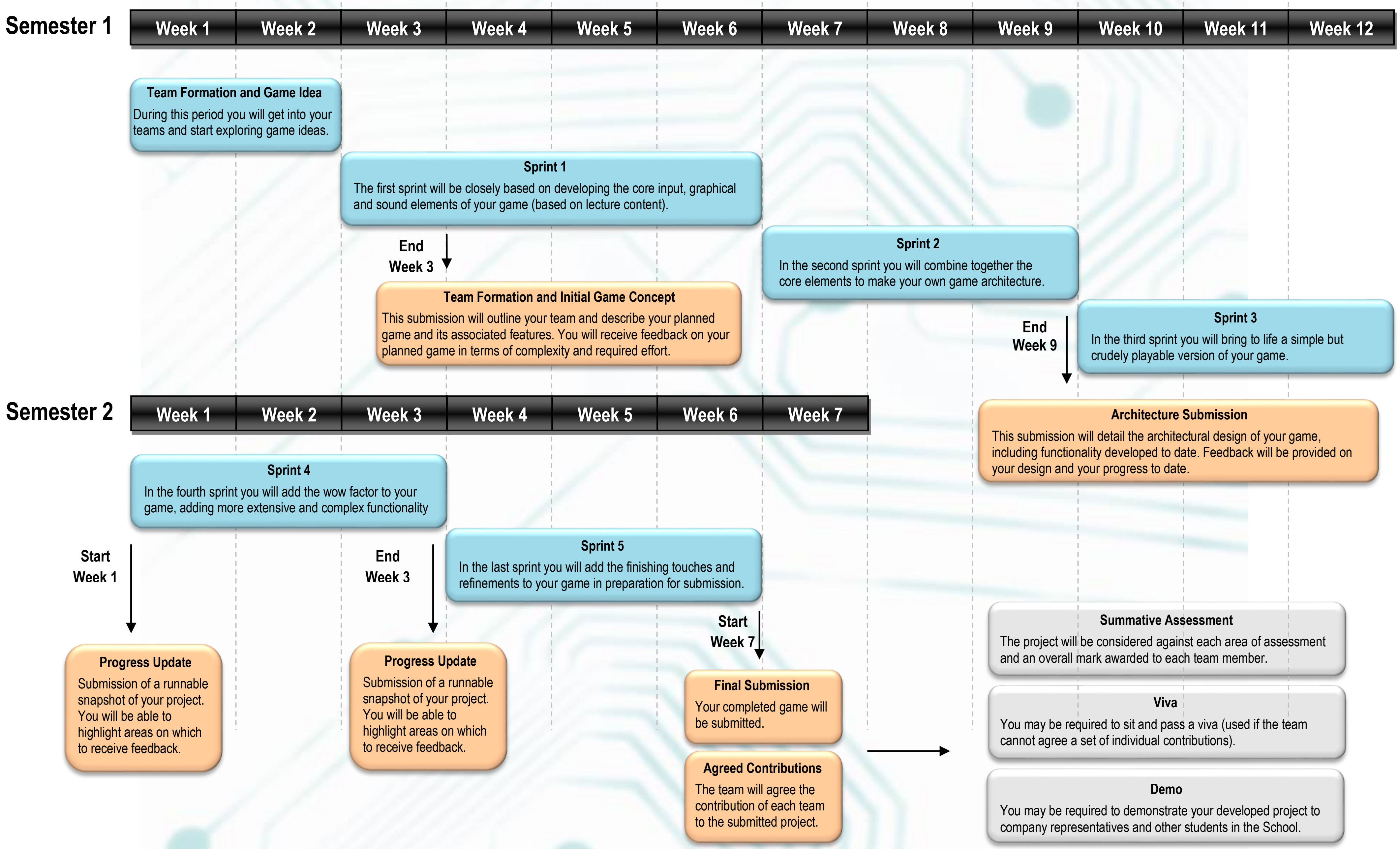
- Firstly, and most importantly, professional programmers will routinely find themselves in a position where they are expected to pick up and learn new technologies. It's par for the course, and the more you do it, the easier it becomes. When you start your placement year you will be expected to learn, mostly under your own steam, a wide range of new technologies.
- One of the key aims of a university level education is to produce graduates who are independent learners – individuals who can pick-up skills and who don't need their 'hand held' to learn. Most third and fourth year modules won't have any practicals – so second year modules offer a mid-way house with a mixture of practical-based and non-practical based modules.

As a counter balance, when marking the project I will fully take into account the effort that was needed to pick up and learn Android programming (i.e. whilst you have been presented with a more significant learning challenge the assessment will be slightly less demanding to take this into account).

The project timeline

A timeline is provided on the next page highlighting project milestones and hand-in points. Please don't worry if the prospect of developing a game appears somewhat daunting, or if the process of how to do so is unclear. As we get further into the module things will become increasingly clear.

Project Structure



Team Formation and Game Idea

During this period you will get into your teams and start exploring game ideas.

Sprint 1

The first sprint will be closely based on developing the core input, graphical and sound elements of your game (based on lecture content).

Team Formation and Initial Game Concept

This submission will outline your team and describe your planned game and its associated features. You will receive feedback on your planned game in terms of complexity and required effort.

Sprint 2

In the second sprint you will combine together the core elements to make your own game architecture.

Sprint 3

In the third sprint you will bring to life a simple but crudely playable version of your game.

Architecture Submission

This submission will detail the architectural design of your game, including functionality developed to date. Feedback will be provided on your design and your progress to date.

Sprint 4

In the fourth sprint you will add the wow factor to your game, adding more extensive and complex functionality

Sprint 5

In the last sprint you will add the finishing touches and refinements to your game in preparation for submission.

Progress Update

Submission of a runnable snapshot of your project. You will be able to highlight areas on which to receive feedback.

Progress Update

Submission of a runnable snapshot of your project. You will be able to highlight areas on which to receive feedback.

Final Submission

Your completed game will be submitted.

Agreed Contributions

The team will agree the contribution of each team to the submitted project.

Summative Assessment

The project will be considered against each area of assessment and an overall mark awarded to each team member.

Viva

You may be required to sit and pass a viva (used if the team cannot agree a set of individual contributions).

Demo

You may be required to demonstrate your developed project to company representatives and other students in the School.

Assessment

How is the project assessed?

Each project will receive a mark out of 100 based on the how well it delivers against the following categories:

- **Professionalism**: To what extent have you approached the development of your game in a professionalism manner? In order to score highly in this section you will need to make effective use of software development tools, such as versioning, alongside delivering a professional project demonstration.
- **Quality of architectural design**: To what extent have you designed the architecture of your game (in terms of objects, classes, interfaces, etc.) in a manner that is appropriate to your proposed game? In order to score highly in this section you will need to ensure that your developed architecture is both extensible and reusable.
- **Use of input, graphics and sound**: To what extent have you used input, graphics and sound to good effect within your game? Marks will be awarded based on the impressiveness of *coding* within your game. The input/graphics/sound should also provide your game with a 'wow' factor – this might include particularly impressive gesture recognition, animations, graphical layering, contextual sounds, etc.
- **Extent of game features**: To what extent have you included lots of features within your game? Features, in this context, will encompass both the core features of your game and any bells or whistles you might add along the way. For example, you might have added a particularly impressive overlay interface to the game, a nice high-score table with persistent top-scores, etc.
- **Complexity of game algorithms**: To what extent have you developed complex algorithms within your game? For example, marks will be awarded for any particularly impressive artificial intelligence, collision detection, path finding, etc. routines that you have developed. Likewise, marks will be awarded to mathematically rich algorithms, e.g. those involving the use of path projections, etc. Simply making use of a readily available, complex, off-the-shelf algorithm will not attract marks unless you have modified/adapted the algorithm within your game.
- **Coding style and code quality**: To what extent is your code well structured, easily readable and understandable, and also readily maintainable and extensible? Additionally, to what extent does your code appropriately exploit Java capabilities and the Android SDK?

The categories identified above will be weighted as follows (out of a total of 100 marks):

Category	**Mark Range**	**Flexibility**
Professionalism	10 marks	Fixed
Quality of architectural design	15 marks	Fixed
Use of input/graphics/sound	5-25 marks	Variable
Extent of game features	5-25 marks	Variable
Complexity of game algorithms	5-25 marks	Variable
Coding style and code quality	25 marks	Fixed

Notice that some categories are fixed, e.g. 25 marks for coding style and quality, while other categories have a range of marks available (i.e. anything from 5 marks up to 25 marks). The reason for this is that I will let you decide how you wish the mark distribution to be tailored to your particular game. For example, if you intend to develop something with lots of nice graphical effects and complex AI but with one small level and no bells and whistles, then you might wish to select 20+ marks for the 'Use of input/graphics/sound' and 'Complexity of game algorithms' sections and minimal marks in the 'Extent of game features' section. In other words, you can tailor the marking scheme to the type of game that you wish to develop.

You will be able to decide how you wish marks to be distributed as part of the final project submission, although, if I believe that your mark distribution does not best match the strengths of your project then I will adjust the distribution as needed on your behalf.

Project assurances

You are free to make use of existing algorithms, code fragments, classes and libraries developed by other programmers in order to support the development of your game. In fact, it is true to say that there is a proverbial treasure trove of code readily available online. This is a good thing, as it means you can avoid reinventing the wheel in many instances and develop a more impressive game as a consequence. However, I am mindful that this also calls into question the authenticity and authorship of any submitted code. This is certainly not a reflection of a lack of trust on your behalf, but rather more a reflection of the importance ascribed to being able to demonstrate the soundness of any given mark.

In terms of how marks are assured on this module, I intend to make use of the following mechanisms:

- Software versioning (SVN) will be used to track the development of the project by each team member.
- The final submitted project will be subject to an on-line plagiarism test. Using other code is not a problem if, and only if, you've referenced the source.

Assurance vivas will be held to assure the authenticity and authorship of any work over which there is uncertainty. Vivas may also be used in borderline cases to decide what category of mark should be awarded, e.g. a 1st class mark or a 2.1. The viva will explore topics relating to both the project and programming.

All suspected academic offences will be subject to an investigation under the University's Academic Offences regulations. Given the weighting of the project, the University will deem any plagiarism within the project as a 'major academic offence' (see Study Regulations within the University Calendar for further information). You will not want to find yourself in this position as a claim of 'I didn't know' simply will not wash.

How to safely reference code

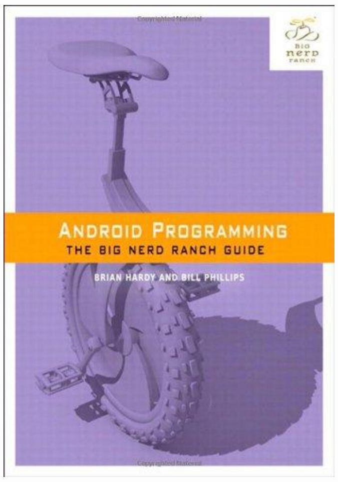
Thankfully, it's easy to safely reference code, as outlined below:

- If making use of a class from another source, then clearly identify the original author within the class header and provide a URL link if possible. Please also include details of any adaptation or modification you've made as you will be given marks based on the extent and manner in which you have reused and/or adapted the class.
- If making use of a code fragment from another source within a method, then clearly identify the original author within the method header and provide a URL link if possible. Again, if you've modified, extended or adapted the source then please include details as you'll gain marks on the extent and manner of any change.

Recommended Reading

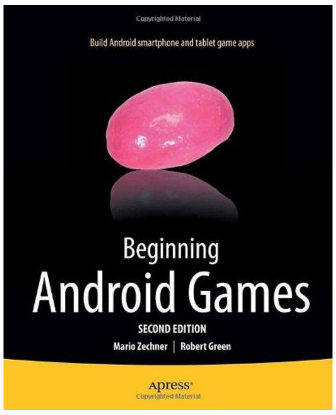
Android Programming: The Big Nerd Ranch Guide
Bill Phillips, Brian Hardy
Paperback: 580 pages
Publisher: Big Nerd Ranch Guides; 1st Ed. (2013)
ISBN-10: 0321804333
ISBN-13: 978-0321804334

This book provides a good introduction to (non-game) Android programming. If you are looking for a broad introduction to app development then do pick it up.



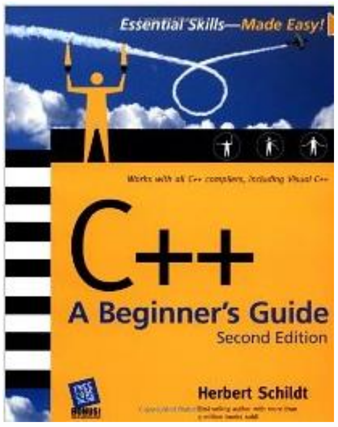
Beginning Android Games Paperback (2nd Ed.)
Robert Green
Paperback: 718 pages
Publisher: Springer; 2nd Ed. (2012)
ISBN-10: 1430246774
ISBN-13: 978-1430246770

This book provides an excellent overview of how games can be developed using Android. It offers a good introduction to OpenGL ES (for hardware accelerated rendering), although, there is little coverage of non-game essential Android APIs


Build Android smartphone and tablet game apps
Beginning
Android Games
SECOND EDITION
Mario Zechner | Robert Green
Apress

C++: A Beginner's Guide (2nd Ed.)
Herbert Schildt
Paperback
Publisher: McGraw-Hill Osborne (2003)
ISBN-10: 0072232153

Whilst this book might purport to be a beginner's guide it does not shy away from providing lots of in-depth coverage and useful advice. Ideal for those who wish to really improve their C++



Contact Info / Links

First Semester Lecture/Advisory Times:

Monday Lecture	10:00 – 11:00	ECS2/0G/006
Tuesday Lecture	13:00 – 14:00	ECS2/0G/006
Thursday Lecture	13:00 – 14:00	ECS2/0G/006
Optional Monday Advisory	14:00-15:00	01 006 / 14 Malone Road
Optional Thursday Advisory	16:00-17:00	01 006 / 14 Malone Road

Second Semester Lecture/Advisory Times:

Tuesday Lecture	14:00 – 15:00	ECS2/0G/006
Wednesday Lecture	10:00 – 11:00	ECS2/0G/006
Thursday Lecture	16:00 – 17:00	ECS2/0G/006
Optional Tuesday Advisory	14:00-15:00	01 006 / 14 Malone Road
Optional Thursday Advisory	15:00-16:00	01 006 / 14 Malone Road

Important Dates:

Friday 17th October	Team formation and Initial game concept hand-in
Friday 28th November	Sprint 2 (Architecture) Submission
Monday 2nd February	Sprint 3 (Progress) Submission
Friday 20th February	Sprint 4 (Progress) Submission
Monday 16th March	Final full game submission
17th/18th/19th/20th March	Assurance vivas

Contact Information:

Philip Hanna
028 9097 4634
Room 01 006, 14 Malone Road
P.Hanna@qub.ac.uk